AI Assignment: Practice with the Basics, Evaluators, and Property Lists

Abstract:

This assignment reinforces basic forms of lisp and works up to more intricate tools like evaluators and property lists. The first task is an assortment of these simpler forms, task 2 and 3 involve creating some functions based around math and distance. Tasks 4 and 5 are two different types of simple calculators someone can build in Lisp. Finally, Task 6 involves property lists and get commands.

Task 1: Simple Things

Task 1: Distance Between Two Points

```
[1]> ( setf x1 6 )
6
[2]> ( setf y1 3 )
3
[3]> ( setf x2 10 )
10
[4]> ( setf y2 13 )
13
[5]> ( setf x-diff ( - x2 x1 ) )
4
[6]> ( setf y-diff ( - y2 y1 ) )
10
[7]> ( setf distance ( sqrt ( + ( expt x-diff 2 ) ( expt y-diff 2 ) ) ) )
10.770329
```

Task 1: Referencing X

```
[1]> ( CADDR '( A B X C D ) )
X
[2]> ( CAADDR '( A B ( X ) C D ) )
X
[3]> ( CADR ( CADAR '( ( A ( B X C D ) ) ) ) )
X
```

Task 1: A Few Meta-Forms

```
[1]> ( setf colors '( red gold blue hazel teal emerald lavender ) )
(RED GOLD BLUE HAZEL TEAL EMERALD LAVENDER)
[2]> ( quote colors )
COLORS
[3]> 'colors
COLORS
[4]> colors
(RED GOLD BLUE HAZEL TEAL EMERALD LAVENDER)
[5]> ( describe 'colors )

COLORS is the symbol COLORS, lies in #<PACKAGE COMMON-LISP-USER>, is
accessible in 1 package COMMON-LISP-USER, a variable, value:
(RED GOLD BLUE HAZEL TEAL EMERALD LAVENDER).

 #<PACKAGE COMMON-LISP-USER> is the package named COMMON-LISP-USER. It
 has 2 nicknames CL-USER, USER.
 It imports the external symbols of 2 packages COMMON-LISP, EXT and
 exports no symbols, but no package uses these exports.

 (RED GOLD BLUE HAZEL TEAL EMERALD LAVENDER) is a list of length 7.
```

```
[6]> ( describe colors )

(RED GOLD BLUE HAZEL TEAL EMERALD LAVENDER) is a list of length 7.

[7]> ( type-of 'colors )
SYMBOL
[8]> ( type-of colors )
CONS
[9]> ( typep colors 'cons )
T
[10]> ( typep colors 'list )
T
[11]> ( typep colors 'symbol )
NIL
[12]> ( typep 'colors 'cons )
NIL
[13]> ( typep 'colors 'symbol )
T
```

Task 1: B5 Deck

```
[1]> ( setf CLUBS '( (10 C) (J C) (Q C) (K C) (A C) ) )
((10 C) (J C) (Q C) (K C) (A C))
[2]> ( setf DIAMONDS '( (10 D) (J D) (Q D) (K D) (A D) ) )
((10 D) (J D) (Q D) (K D) (A D))
[3]> ( setf HEARTS '( (10 H) (J H) (Q H) (K H) (A H) ) )
((10 H) (J H) (Q H) (K H) (A H))
[4]> ( setf SPADES '( (10 S) (J S) (Q S) (K S) (A S) ) )
((10 S) (J S) (Q S) (K S) (A S))
[5]> DIAMONDS
((10 D) (J D) (Q D) (K D) (A D))
[6]> ( setf DECK ( append CLUBS DIAMONDS HEARTS SPADES ) )
((10 C) (J C) (Q C) (K C) (A C) (10 D) (J D) (Q D) (K D) (A D) (10 H)
 (J H) (Q H) (K H) (A H) (10 S) (J S) (Q S) (K S) (A S))
[7]> DECK
((10 C) (J C) (Q C) (K C) (A C) (10 D) (J D) (Q D) (K D) (A D) (10 H)
 (J H) (Q H) (K H) (A H) (10 S) (J S) (Q S) (K S) (A S))
[8]> ( setf PARTITIONED-DECK ( list CLUBS DIAMONDS HEARTS SPADES ) )
(((10 C) (J C) (Q C) (K C) (A C)) ((10 D) (J D) (Q D) (K D) (A D))
 ((10 H) (J H) (Q H) (K H) (A H)) ((10 S) (J S) (Q S) (K S) (A S)))
[9]> PARTITIONED-DECK
(((10 C) (J C) (Q C) (K C) (A C)) ((10 D) (J D) (Q D) (K D) (A D))
 ((10 H) (J H) (Q H) (K H) (A H)) ((10 S) (J S) (Q S) (K S) (A S)))
```

Task 2: Distance Between Two Points

Distance Between Two Points Code

```
3  ( defun distance (x1 y1 x2 y2)
4      ( sqrt ( + ( expt (- x2 x1 ) 2 ) ( expt (- y2 y1) 2 ) ) )
5  )
```

Distance Between Two Points Demo

```
[2]> ( distance 1 2 3 4 )
1.4142135
[3]> ( distance 4 3 2 1 )
1.4142135
[4]> ( distance 10 12 8 10 )
2.828427
[5]> ( distance 10 8 12 10 )
2.828427
[6]> ( distance 5 10 15 20 )
7.071068
```

Task 3: Perimeter of Triangle

Perimeter of Triangle Code

```
 7   ( defun perimeter (x1 y1 x2 y2 x3 y3)
 8       ( +
 9           ( distance x1 y1 x2 y2 )
10           ( distance x1 y1 x3 y3 )
11           ( distance x2 y2 x3 y3 )
12       )
13   )
```

Perimeter of Triangle Demo

```
[8]> ( perimeter 1 2 3 4 5 6 )
11.313708
[9]> ( perimeter 10 10 10 10 10 10 )
0
[10]> ( perimeter 0 0 5 0 0 5 )
17.071068
[11]> ( perimeter 5 0 10 10 0 5 )
29.431747
[12]>
( perimeter -5 0 5 5 -5 5 )
26.18034
```

Task 4: LR Infix Calculator

LR Infix Calculator Code

```
( defun calculator-LR (&aux input operators operands partial-result result)
   ( format t "Expression? " )
   ( setf input ( list (read) (read) (read) (read) (read) ) )
   ( setf operators ( list ( second input ) ( fourth input ) ) )
   ( setf operands ( list ( first input ) ( third input ) ( fifth input ) ) )
   ( setf partial-result ( apply ( first operators) ( list  ( first operands ) ( second operands ) ) ) )
   ( setf result ( apply ( second operators ) ( list partial-result ( third operands ) ) ) )
   ( format t "Result: ~A~%" result)
   ( calculator-LR )
)
```

LR Infix Calculator Demo

```
[2]> ( calculator-LR )
Expression? 12.3 + 4.5 * 9.6
Result: 161.28
Expression? 12.3 / 4.5 - 9.6
Result: -6.866667
Expression? 14.5 / 8.2 * 3.3
Result: 5.835366
Expression? 44.4 - 33.3 / 11.1
Result: 1.0000001
Expression? 13 / 13 - 1
Result: 0
Expression?
```

Task 5: RL Infix Calculator

RL Infix Calculator Code

```
( defun calculator-RL (&aux input operators operands partial-result result)
   ( format t "Expression? " )
   ( setf input ( list (read) (read) (read) (read) (read) ) )
   ( setf operators ( list ( second input ) ( fourth input ) ) )
   ( setf operands ( list ( first input ) ( third input ) ( fifth input ) ) )
   ( setf partial-result ( funcall ( second operators ) ( second operands ) ( third operands ) ) )
   ( setf result ( funcall ( first operators ) ( first operands ) partial-result ) )
   ( format t "Result: ~A~%" result)
   ( calculator-RL )
)
```

RL Infix Calculator Code Demo

```
[2]> ( calculator-RL )
Expression? 12.3 + 4.5 * 9.6
Result: 55.5
Expression? 12.3 / 4.5 - 9.6
Result: -2.4117646
Expression? 14.5 / 8.2 * 3.3
Result: 0.5358463
Expression? 44.4 - 33.3 / 11.1
Result: 41.4
Expression? 13 / 13 - 1
Result: 13/12
Expression?
```

Task 6: Representing Quarto Pieces using Property Lists

Representing Quarto Pieces using Property Lists Code

```
( setf *pieces* '(BRHS BRHC BRSS BRSC BBHS BBHC BBSS BBSC SRHS SRHC SRSS SRSC SBHS SBHC SBSS SBSC) )

( setf ( symbol-plist 'BRHS ) '( Size big color red style hollow shape square ) )
( setf ( symbol-plist 'BRHC ) '( Size big color red style hollow shape circle ) )
( setf ( symbol-plist 'BRSS ) '( Size big color red style solid shape square ) )
( setf ( symbol-plist 'BRSC ) '( Size big color red style solid shape circle ) )
( setf ( symbol-plist 'BBHS ) '( Size big color blue style hollow shape square ) )
( setf ( symbol-plist 'BBHC ) '( Size big color blue style hollow shape circle ) )
( setf ( symbol-plist 'BBSS ) '( Size big color blue style solid shape square ) )
( setf ( symbol-plist 'BBSC ) '( Size big color blue style solid shape circle ) )
( setf ( symbol-plist 'SRHS ) '( Size small color red style hollow shape square ) )
( setf ( symbol-plist 'SRHC ) '( Size small color red style hollow shape circle ) )
( setf ( symbol-plist 'SRSS ) '( Size small color red style solid shape square ) )
( setf ( symbol-plist 'SRSC ) '( Size small color red style solid shape circle ) )
( setf ( symbol-plist 'SBHS ) '( Size small color blue style hollow shape square ) )
( setf ( symbol-plist 'SBHC ) '( Size small color blue style hollow shape circle ) )
( setf ( symbol-plist 'SBSS ) '( Size small color blue style solid shape square ) )
( setf ( symbol-plist 'SBSC ) '( Size small color blue style solid shape circle ) )

( defun display-random-piece ()
   ( setf choice (nth ( random 16 ) *pieces* ) )
   ( setf properties (list choice ( get choice 'size ) ( get choice 'color ) ( get choice 'style ) ( get choice 'shape ) ) )
   properties

)
```

Representing Quarto Pieces using Property Lists Demo

```
[2]> *pieces*
(BRHS BRHC BRSS BRSC BBHS BBHC BBSS BBSC SRHS SRHC SRSS SRSC SBHS SBHC SBSS SBSC)
[3]> ( symbol-plist 'brsc )
(SIZE BIG COLOR RED STYLE SOLID SHAPE CIRCLE)
[4]> ( symbol-plist 'sbhs )
(SIZE SMALL COLOR BLUE STYLE HOLLOW SHAPE SQUARE)
[5]> ( display-random-piece )
(BBSS BIG BLUE SOLID SQUARE)
[6]> ( display-random-piece )
(SRSC SMALL RED SOLID CIRCLE)
[7]> ( display-random-piece )
(SRSS SMALL RED SOLID SQUARE)
[8]> ( display-random-piece )
(SBSS SMALL BLUE SOLID SQUARE)
[9]> ( display-random-piece )
(SBSS SMALL BLUE SOLID SQUARE)
[10]> ( display-random-piece )
(BRHC BIG RED HOLLOW CIRCLE)
```